REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	30 APR 1997	**Final Technical Report (Sep '96 - Apr '97)**

4. TITLE AND SUBTITLE

System Synthesis Environment

5. FUNDING NUMBERS

C - DAAH01-96-C-R298

PR - PAN RTW X8-96

6. AUTHORS

Ramesh M. Reddi **Infopike, Inc.**

Prof. Reda Ammar **Univ of Connecticut**

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Infopike, Inc., P. O. Box 328, Norwich, CT 06360

University of Connecticut, Storrs, CT 06269

8. PERFORMING ORGANIZATION REPORT NUMBER

8. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This is the Final Technical Report of the SBIR Phase I contract awarded by DARPA. The project deals with an Architecture Assesment Environment approach to assist engineers in the performance modeling and analysis of complex software systems. The role of analytical performance modeling in the software design environment is stressed over simulation or measurement.

A list of technical objectives and a detailed system for Phase II SBIR work is described. An architecture to implement the Hierarchical Performance Modeling (HPM) is proposed. The architecture is comprised of an Object-oriented database; tools to model performance at operational, task, module and system levels; Facilities to construct models at system, task, module and operational levels; and a graphical user interface (GUI) to interface with all the components of the system.

A brief account of the industrial consultations that have been planned and in progress is given. Two categories of applications are being investigated: a) High Performance Computing and b) Real-Time embedded computing.

14. SUBJECT TERMS

System Synthesis, High Performance Modeling, System level, Task level, module level,

Queuing model, task model, computational structure model, real-time systems

15. NUMBER OF PAGES
32

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	**UNCLASSIFIED**	**UNCLASSIFIED**	**SAR**

Infopike, Inc.

Final Technical Report

Apr 30, 1997

Sponsored by

Defense Advanced Research Projects Agency
(DARPA - ITO)

ARPA Order Nr. D611, Amdt 05

Issued by U. S. Army Missile Command Under

Contract No. DAAH01-96-C-R298

Name of Contractor	**Infopike, Inc.** P. O. Box 328 Norwich, CT 06360	Ph: 860-464-7990 Fax: 860-464-7980 WEB: http://www.infopike.com
Principal Investigator	Ramesh M. Reddi E-mail: ramesh@infopike.com	

Contract Dates	Effective Date	Sep 19, 1996
	Expiration Date	Apr 30, 1997
	Reporting Period	Sep 19, 1996 - Apr 30, 1997

System Synthesis Environment

CONTENTS

A. Project Background and Objectives

Infopike, Inc. was awarded a SBIR (Small Business Innovation Research) Phase I contract by DARPA (Defense Advanced Research Projects Agency) in Sep '96. The contract number is DAAH01-96-C-R298. The contract requires Infopike to do research on the topic "System Synthesis Environment." The performance period for Phase I is six months. Infopike teamed up with University of Connecticut to execute this contract. Infopike is required to submit the Final Report to DARPA as part of the contract deliverables. This is the Final Report of this SBIR Phase I contract.

The use of parallel/distributed architectures to alleviate what is typically referred to as the "Von Neumann bottleneck" has long been a goal. Increasingly complex problems such as those required by military systems, "grand challenge" applications, etc., demand the increased utilization of parallel and distributed architectures. Unfortunately, designing and assessing the performance of software systems for parallel/distributed architectures is difficult, and this is compounded when the application has real-time performance requirements. The topological structures that are used can have significant variation, and range from hypercubes, meshes, etc., to the now-popular Commercial Off-The-Shelf (COTS) clusters of PCs and workstations, with a variety of new high-speed communications technology (ATM, High-speed Ethernet, FDDI, etc.)

The tools available to assist in integrating the software architecture and processor topology resulting in candidate system designs are limited. This set is further reduced when high degrees of parallelism are required and/or real-time performance constraints are imposed. In addition, many of the tools (e.g., compilers) were developed just to incorporate facilities to express parallelism, or distribution, and leave the system optimization entirely in the hands of the designer. Tools to assist the designer in this area are now just beginning to emerge; many are based on the use of performance traces to allow a basis for adjustment after a design has been completed. Counter to this, our approach integrates the solution to the performance analysis/optimization into the design and development process.

Therefore, there exists a need in industry for an ARCHITECTURE ASSESSMENT ENVIRONMENT product that can support the expression of: processor and communications topology; software system structure; and performance requirements of the overall composite under different workloads. The term "architecture" in this context refers to the composite hardware and software structure. This environment would allow an initial performance-sensitive design to be developed to form a solid basis for the real test and evaluation process. We expect that this approach would add time and effort to the initial design/development period, but would simplify the test and evaluation, reduce the costs of redesign and maintenance, and produce a higher sensitivity and capability for this type of task in the mind of the designer. In addition, while the design process is emphasized, the environment could also be used to evaluate systems that have already been designed and/or implemented in order to help identify existing limitations and characterize their behavior.

The ARCHITECTURE ASSESSMENT ENVIRONMENT should be capable of utilizing various hierarchical levels of system abstraction in order to support analysis of complex hardware/software composite systems. Further, the environment must incorporate sufficient expressive power to support the required functionality using tools and techniques best suited for particular modeling application. This indicates that the environment must be developed as an "open" system to allow the addition, replacement of tools to carry out each of the primary steps of the design/development

process. Analysis must be computer-automated and must support the ability to incorporate measures of performance effectiveness that are both generic and application specific.

In phase I we addressed the specification of the SYSTEM SYNTHESIS ENVIRONMENT which would provide Hierarchical Performance Modeling (HPM) capabilities in a commercial performance assessment toolset. The main effort in phase I was to specify the properties and feasibility of such a toolset and to understand its potential application in the commercial sector. These specifications include: specification of a data base for the performance tool, specification of the Hierarchical Performance Modeling (HPM) methodology, specification of a graphical user interface, specification of the application focus area, and development of a marketing strategy. In addition, an industrial review was required to discuss the proposed tools and capabilities with a selected set of customers known to Infopike. The collective comments and opinions regarding performance modeling of commercial computer systems and applications was added to the data from Phase I.

B. Phase I Project Activities

We report the results of our research investigation in the following areas:

- Technical Survey
- System Specifications
- Industrial Consultations/Review
- Review Meetings
- Internet Marketing
- Software Base Selection
- Modeling Specification
- User Interface
- Open System Mechanism
- Application Environment
- Tool Set Selection
- Product Evaluation
- Training
- Case Studies

B.1 Technical Survey

A detailed technical survey was performed to get insight on the performance modeling. The effort involves a study of at least 235 research sources from conference papers, journal articles, web sites, technical reports and so forth. A compilation of this in-depth study is made as a separate report. Some of the key findings of this survey are given in this report. The report is being submitted along with this Final report.

B.1.1 Performance Modeling Approaches

From a high level perspective, there are three principal methods for computer performance estimation and analysis: 1) direct measurement; 2) simulation; and 3) analytic modeling. These

methods do not have to be used in isolation, and it is normal to use a combination of two or more during construction of a performance model. Within each of these categories there are a number of techniques available for modeling parallel and distributed systems.

B.1.1.1 Direct Measurement

Direct measurement is the process of collecting performance information by observing the system as it runs. For estimating a parameter's value, direct measurement may involve hardware/software monitoring, data collection and statistical analysis. For software monitoring, measurement is conducted by placing probes into the source code that can monitor the time required by critical software components.

Measurement is more accurate than other types of modeling but also the least useful for system design purposes. This is primarily because the designer must completely implement a hardware/software design in order to collect system timing information. The problems exposed through measurement can only be corrected by re-design and re-development of the hardware or software. Re-design of an entire hardware/software system is a very costly process. In addition, measurement alone does not give any guidance on how to redesign the system. In many respects, measurement is a technique which is applied too late in the design cycle to be useful by itself. The area where measurement can be extremely useful is in verification of an existing performance model, and in obtaining the parameters required to build a simulation or analytic model.

B.1.1.2 Simulation

Simulation is commonly applied to many of the performance modeling problems being confronted in parallel and distributed systems. Simulation is accomplished by creating a virtual system which imitates the discrete events within the real, physical system. The ability to re-create discrete behaviors is a primary advantage, because some system behaviors/properties are outside the boundaries of analytic modeling. The simulation can be accomplished by various techniques such as Monte Carlo simulation, trace-driven simulation, and discrete-event simulation.

The performance measures supplied by a simulation are fairly accurate, provided the system can be modeled at a high level of detail. Unfortunately, simulation models are difficult to construct and time consuming to run and validate. If a parallel/distributed system is simulated at an adequate level of detail, the simulation run-time or development time may be so long as to be unacceptable. The problem is compounded by the multiple design alternatives, each of which requires additional simulation runs at each parameter value. In addition, it is often difficult to find insights using the data supplied from simulation experiments. Insights into the relationships between performance parameters are much easier to discover using an analytic approach which generates a performance equation.

B.1.1.3 Analytic Modeling

Analytic modeling comprises various well defined techniques such as queuing models, Markov models, petri-nets, statistical analysis and computation structure models. Analytic modeling is considered the least "expensive" modeling approach for two reasons:

- Complete hardware/software implementation is not required as it is for most monitoring approaches.

- Simulation design, programming, testing and debugging are more time consuming than using standard analytic models.

The "expense" of performance analysis is defined in terms of *time and effort* required to build the model and analyze the results. When compared to simulation, analytic models are less time consuming to run and validate. As another advantage, analytic models furnish insights into the relationships and interactions between variables which are difficult to uncover with measurement or simulation techniques.

Analytic models do have limitations. For some problems, the assumptions used in the analytic model do not match the real system's behavior. Also, due to the intricacies of the operations within a parallel/distributed system, no analytic model can represent everything that goes on inside the real system. Therefore most analytic models incorporate the dominant system operations and ignore the rest. As expected, every operation that is ignored causes the analytic model to incur an unknown degree of error.

B.1.2 A Sample of Available Performance Tools

NASA Ames Research Center developed a tool called AIMS (Automated Instrumentation and Monitoring System). AIMS models parallel programs by using simulation-based and analytical-based approaches and automates the process of building performance prediction models. NASA Ames Research Center also uses a tool ATEXPERT that was developed by CRAY. ATEXPERT monitors performance and displays measurement data graphically. The tool works well on CRAY and SGI architectures.

The Technical University of Munich developed a performance prediction environment, APNM (Abstract Parallel Numerical Machine). It profiles the programs and their performance is predicted through an analytical model. The University of Vienna, Austria, developed VFPMS (Vienna Fortran Performance Measuring System). VFPMS serves as an analysis tool for their FORTRAN Compiler. However, the tool does not generate a model. The University of Vienna developed another tool PEPSY (PErformance Prediction SYstem) that generates a simulation model of the program under scrutiny.

Oak Ridge National Laboratory developed a tool called ParaGraph that does a trace-based performance visualization of message-passing parallel programs. There upon Intel Supercomputer Systems Division developed ParaGraph-based tool set that yields graphical summaries and statistical analysis of overall program behavior. INTEL recently developed a tool called VTUNE to analyze C and FORTRAN programs on Windows 95 and Windows NT platforms.

NEC European Supercomputer Systems in Switzerland developed the tool Annai. It is used for performance analysis of software programs at different abstraction levels. Argonne National Laboratory developed a tool called UPSHOT that analyses trace executions produced by parallel programs and includes a graphical tool for visualizing parallel program behavior.

B.1.3 Abstraction Levels

Within the categories of measurement, simulation and analytic approaches, performance techniques can be classified in terms of the level of analysis which they provide. Parallel and distributed systems are normally composed of three or more abstraction levels which correspond to

logical boundaries in the design of hardware, application software and operating system services. To classify different types of performance models, a parallel or distributed system is decomposed into a series of modeling levels. Each level is a different "view" of the system at a particular hardware or software boundary. The four basic levels are: 1) System Level; 2) Task Level; 3) Module Level; 4) Operation Level.

The operation level provides the lowest level of abstraction for computer performance modeling. The operation level estimates the time cost of primitive operations (such as addition, multiplication, assignment, etc.) which are contained in a software module. This level of analysis is determined by the high-level language compiler and microprocessor combination which establishes primitive time costs. Examples of this type of analysis are given by work done in the area of compiler/microprocessor simulation and modeling.

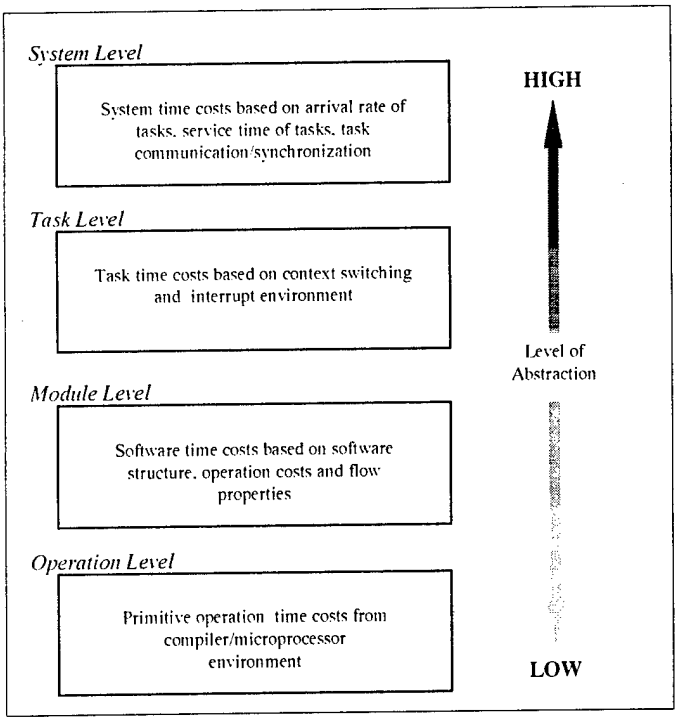

Figure 1 Performance Modeling Levels

The module level resides above the operation level in the series of levels. The module level calculates the time costs for segments of the software program based on the control structures specified by the source code of a high level language. Examples of this type of analysis are given by work done in the area of software execution graphs and computation structure models.

At the task level, the analysis is based on the interaction of software modules which are running concurrently. At this level, the system is viewed as a set of independent software units (e.g. tasks or threads) which synchronize and communicate in order to complete a larger job. Since the interaction between tasks changes the expected execution time of a software module, the task level analysis must provide estimates of the slowdown caused by interrupts and context switching.

System level analysis describes a parallel or distributed application at the highest level of abstraction utilizing coarse-grained estimates of the most significant service times, arrival rates and

resource properties. Frequently the complete computer system is modeled as a single entity such as a queuing network. Queuing models become necessary whenever a contention for resources causes tasks to wait in a buffer for service. The queuing paradigm is typically found in several components of a parallel or distributed system, in particular within the operating system, and in communications.

B.1.4 Hierarchical Performance Modeling (HPM) Methodology

Parallel software performance research includes the development of software execution models and tools which can derive the detailed (micro) time cost of an application. Tools were developed to generate analytic time cost equations by parsing the source code of a Dataparallel C application. Researchers at Syracuse University have developed an environment for analytic performance prediction of FORTRAN 90D code using the Intel iPSC/860 as a hardware platform. The environment integrates a HPF/FORTRAN 90D compiler, a functional interpreter, a source-based performance prediction tool and a graphical user interface.

PEPS (Performance Evaluation of Parallel Systems) is a Mathematica-based tool and framework for performance prediction which is being developed by researchers at the University of Warwick, United Kingdom. The PEPS framework allows the designer to analyze performance using a software execution graph and a set of analytic/simulation methods in a layered model.

The work being done by Connie Smith in Software Performance Engineering (SPE) is also relevant, because it encourages system designers to assess performance using analytic techniques in the early stages of the software life cycle. Further details of the PEPS and SPE approaches are given in the section on related work.

B.1.4.1 Properties of the Hierarchical Performance Modeling (HPM)

This section compares the HPM to other methodologies. Currently there are two primary methodologies, SPE and PEPS, which address some of the same needs as the HPM. The SPE and PEPS methodologies also share the following properties:

- Analytic models are used to assess time-cost performance.
- A general-purpose methodology is proposed to model parallel/distributed applications.
- The methodology incorporates the detailed software structure as part of the performance model.

In order to relate to SPE and PEPS, three areas are shown below which list the important properties of the Hierarchical Performance Modeling (HPM).

Basic Area:

1. Well defined, general-purpose methodology for modeling parallel/distributed systems.
2. Methods for analytic modeling of large, complex software/hardware systems.
3. Analysis provided for System, Task, Module and Operation levels.
4. Methods for model development in software life-cycle integrated w/ SPE.
5. Hierarchical construction with flexibility to match models to design components.
6. Methods to link different types of analytic models at the four levels of abstraction.

Software Area:

1. Provides complete flow/operation analysis of software micro-time cost.
2. Method & tool to derive flows & CFG (control flow graph) from source code.
3. Method & tool to flow balance, calculate dependent/independent flow relationships.
4. Method & tool to generate analytic time cost expression from CFG and operation level primitives.
5. Analytic equations can be related back to software structure for isolation of components.
6. Automated calculation of operation costs based on compiler/microprocessor combination.
7. Methods for reduction of complexity at the operation and module levels.

Practical Area:

1. Methods based on realistic assumptions about information designer can provide.
2. Best-case, worst-case, mean, variance, minimum and maximum calculations.
3. Special processing for designer's uncertainty levels.
4. Special processing for sensitivity and time cost within operating regions.
5. Special processing for reduction/handling of model complexity in large systems.
6. Methods for definition of workload and environment parameters.
7. Automated tools to relieve burden of manual processing of software source code.
8. Automated tool to relieve burden of manual mathematical calculations and analysis.

B.1.4.2 Relation to SPE

SPE (Software Performance Engineering) is a comprehensive methodology for performance analysis throughout the software life-cycle. Currently SPE is the most general methodology which has gained a wide acceptance in the performance research community. The software execution model in SPE is similar to the computation structure models which have been utilized to derive the detailed time cost of software structures in the HPM. For these reasons, it was decided early on in the development of the HPM to enhance some aspects of the SPE methodology, rather than define entirely new approaches. The following goals motivated the development of SPE:

- *Supplement Existing Methods*: Interface with current software development processes to maximize SPE's adoption potential.
- *Provide Ease of Use*: Application of SPE in the early design stages should require less effort than fixing performance problems latter in the implementation and testing stages.
- *Provide Rapid Assessments*: Designers must assess performance in sufficient time to influence software design/development decisions.
- *Provide Sufficient Precision*: Supply enough precision to distinguish acceptable design concepts from potentially disastrous ones and to identify software components whose performance is critical.

- *Use a Life-cycle Paradigm*: Support performance assessment at the analysis, design, implementation and maintenance stages of the software life-cycle.

- *Provide Wide Applicability*: The SPE methodology utilizes models which are generic enough to be applied in most applications including real-time systems, high performance computing systems and client-server systems.

- *Insure Adaptability*: The methods and tools can adapt to the changing needs which are brought on by new hardware and software technologies.

Many of these goals have been reached; however there are missing components, such as adequate support tools and modeling at all four levels.

B.1.4.3 Relation to PEPS

PEPS (Performance Assessment of Parallel Systems) is currently the approach which is most closely related to the research effort described in this report. The PEPS methodology is based on a set of "layers", at which different components of time cost are assessed and different types of modeling approaches are used. The PEPS layers are not organized to directly correspond to the design levels of abstraction, which represents the major difference between PEPS and the Hierarchical Performance Modeling (HPM). The PEPS layers are:

1. Application Layer: This layer models the high level software application using a software execution graph model similar to that applied in SPE. The graph models the flow of control throughout the program and uses different node types to describe sequential code segments, branch operations, loop operations, fork-join operations, lock-unlock operations and sub-task invocation.

2. Sub-task Layer: This layer models the low level software time costs based on sub-tasks and sequential components from the application layer. The sub-tasks are procedures or functions which are invoked at the application layer and may contain parallel constructs. Sequential components of the software are broken down into their primitive time costs at this layer.

3. Parallel Template Layer: This layer is used to combine all the information from the other layers into a composite model which generates the performance metrics. The designer must classify the algorithms of interest into one of the standard classes (such as master-slave, divide and conquer, pipeline, etc.). Each different class of algorithm is evaluated based on its precedence relationships and communication patterns when mapped onto the target hardware platform.

4. Hardware Layer: This layer uses a hierarchical structure to organize the information about an architecture into static and dynamic parameters. Static parameters are those which are not influenced by the run-time environment and can be determined from benchmarking and configuration information. Dynamic parameters, which are influenced by the run time environment, are determined by hardware characterization and analytic modeling.

The PEPS methodology does not currently contain the detailed analytic techniques for analysis of a large, complex application. The layered structure does not have as much flexibility as the HPM, because it cannot always be "molded" to match the design view of the system under study. Although the fixed structure of PEPS is less flexible than the HPM, it has the advantage that it is easier to implement and support as a toolset than the HPM tools. The PEPS standardized layers

and algorithm classes will be harder to match up to a particular system, but will require less intervention from the designer in building performance models.

B.1.4.4 Relation to TCAS

The HPM builds on several years of previous work done at *University of Connecticut*, in the development of the CSM (computation structure model). Based on the CSM, the Time Cost Analysis System (TCAS) was developed to automate derivation of software time cost expressions. TCAS does not provide a general-purpose methodology for modeling parallel/distributed applications. However it does provide performance estimation for single-module programs. In this sense, TCAS is a predecessor of MTED (Module Time-cost Expression Deriver) which is part of the HPM toolset. Fig 2 shows this relationship of the tools.

B.1.4.5 Limitations of the Hierarchical Performance Modeling (HPM)

Several limitations of the HPM are unavoidable because they represent the problems of any modeling approach using a general framework instead of a specific framework. For example, the PEPs tools and the SPE approach have similar limitations to the ones described here for the HPM. The primary limitation of the Hierarchical Performance Modeling (HPM) is the designer involvement required to construct a model. Currently, model construction will require the designer to be knowledgeable in the use of the basic models and how basic models can be applied appropriately.

Designer intervention is also required in order to parameterize the models with the required performance data for the system's workload and run-time environment. This task is required of any modeling approach and does not, by itself, represent a major limitation. Specification of parameters is the most time consuming at the module level, where the designer is required to enumerate the flow properties of the software components. Other modeling approaches, which have targeted specific algorithms, architectures or compilers do not require the same level of intervention from the designer because a large component of the system model is pre-specified before the designer begins the analysis.

Since the HPM relies on analytic modeling techniques, it has all the advantages and disadvantages of analytic modeling. One specific area that may be a concern is the validity of assumptions, in particular for queuing models. In the area of queuing models, there may be an advantage to using the simulation approach if the standard queuing assumptions do not apply. HPM can provide a simulation using the model in place of the physical system.

B.2 System Specifications

The functional organization of the performance tool is shown in figure 2. The designer will work with the graphical user interface to indirectly invoke each of the required tools and facilities to construct and analyze a hierarchical model. Performance information and the hierarchical structure are stored in the object-oriented database. Mathematica support is required at various points for symbolic math processing or generation of plots and graphs.

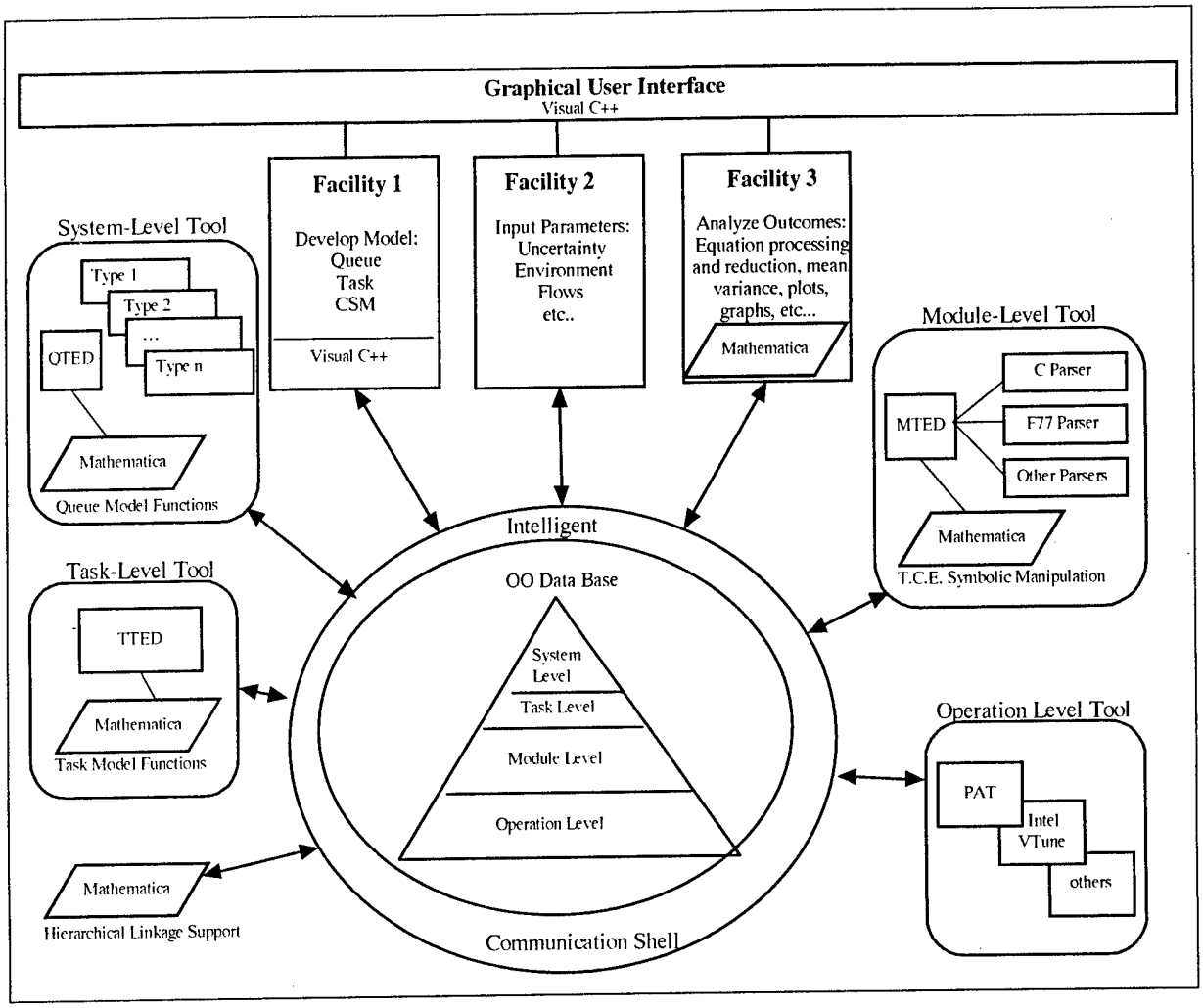

Fig 2. Architecture of proposed performance tool

The major components of the system are as below:

- Operation Level Tool Set
- Module Level Tool Set
- Task Level Tool Set
- System Level Tool Set
- Graphical User Interface
- Facility 1 - Model Development
- Facility 2 - Input Parameters
- Facility 3 - Analysis Outcomes
- Intelligent Communication Shell
- Object-oriented Database

The *operation level tool* component computes time estimates for operation level primitives. The inputs to this component are parameters related to run time environment and experimental measurements. This component generates a table of time cost estimates (min, max, mean and

variance) for symbolic primitive operations for each target processor/compiler. It will also provide a reduced primitive set for each processor/compiler. We currently have PAT (Primitive Assessment Tool) that automatically calibrates an operation level model to fit the selected compiler/processor environment by timing all the required logical primitives. We have recently acquired Intel's VTUNE software and plan to use it in conjunction with our operation level tool.

The *module level tool* derives time cost expression from provided source code and flow parameter information. The inputs are application source code; flow parameter definitions in source code; and the table of symbolic names for current target processor/compiler. The tool generates a time cost expression for each module within the source code; and equations retain calling structure of the original application. We have MTED (Module Time-cost Expression Deriver) that generates time cost equations from source code using the computation structure model as a theoretical basis.

The *task level tool* provides analysis of task models which represent the application at the task level - more specifically this tool calculates effect of interrupts and communication on task execution time. We plan to develop TTED (Task-level Time-cost Expression Deriver) that works with task level models.

The *system level tool* provides analysis of queuing models which represent the application at the system level. We plan to develop QTED (Queuing Time-cost Expression Deriver) that works with queuing models. We intend to consider open and closed models; and exact and approximate solution approaches.

Facility 1 will require some type of "input language" for describing queuing networks and other graph based performance models. Facility 2 will require an interface for input of the parameters of each workload scenario. There are several different types of parameters required at each of the four modeling levels. Facility 3 will provide analysis of the hierarchical model including uncertainty, complexity reduction, conditional upper bound, conditional lower bound, average, variance and other metrics based on the workload scenarios specified with Facility 2. We currently have ASAP (Analytic Software Analysis and Prediction system) which allows designers to construct and analyze complex hierarchical models for software applications. We intend to integrate ASAP into these Facilities.

A graphical user interface (GUI) will be designed to connect all the tool sets, facilities, object-oriented database, intelligent communication shell and Mathematica. We have investigated various GUI platforms and plan to use Visual C++. We believe that Visual C++ can interface best with the existing C++ code and to be developed object-oriented database.

The details of inputs, outputs, and function of some of the above mentioned components are still being developed. The details of the object-oriented database are described in a different section of this report.

B.3 Industrial Consultations/Review

Infopike, along with Univ. of Connecticut, gave presentations to Industrial Community to understand their needs in the Performance Analysis area. The results of the Phase I SBIR research work indicate both a positive interest in hierarchical performance modeling, and a clear path to design/ implementation of a performance prediction tool set.

We have given presentations to Digital Equipment Corporation, IBM Watson Research Laboratories, United Technologies, Analysis & Technology and Pandrol Jackson. We apprised them of the research work done by Infopike-UCONN in the performance modeling area. We are also in the process of giving presentations to defense contractors like Raytheon and Sonalysts. We are also in touch with a number of other organizations like Intel and communicated our goals of commercializing this research effort. The feedback from the Industrial contacts during Phase I SBIR work indicate a high demand for performance modeling in both the high-performance computing domain (Digital Equipment Corporation, Analysis & Technology, IBM) and in the real-time embedded domain (United Technologies and Pandrol Jackson)

We learned from these Industrial Consultations that our SBIR work is very important in the fields of High Performance Computing and Real-time Computing. We plan to develop products to address the needs and requirements in these areas. This effort also helped us to plan our marketing strategy to define our target market that will be in need of our products.

B.4 Review meetings

From the outset, we have organized the project planning through weekly meetings. The team comprising members from Infopike, Inc. and the University of Connecticut meet every week and discuss the progress of the project tasks. The members are assigned weekly tasks based on the research goals stated in our SBIR Phase I proposal to DARPA. At each meeting, the members report their progress and any issues that need to be resolved will be discussed. The senior members of the research team suggest action items for the group to achieve the research goals. Some times guest researchers and professionals are invited to attend these meetings.

B.5 Internet Marketing

This task is to study the feasibility of methods of software delivery and eventual marketing of the product to be developed.

We have decided that the system that needs to be developed should be released in three versions:

- A stand-alone version that runs on Windows NT, Windows 95 and Unix operating systems
- A Client/server or network version with server-version on Windows NT and Unix and client-version on Windows 95 and NT.
- A Web-Based tool using JAVA and Netscape for remote Internet accesses. This assumes that there is a version on the WEB server that can be accessed through thin clients such as Network Computers and PCs.

B.6 Software/Knowledge Base Selection

We are designing an Object-oriented Database and an Intelligent Communication Shell (See Figure 2) that acts as the backbone of the performance tool. The object-oriented database provides a centralized facility for storage of performance related data, including the following:

- Hierarchical structure of the overall model
- Representations of sub-models at each level (graphs, tables, time cost expressions, etc.)

- Parameters, measurement data, statistical information.
- Hardware configurations
- Software configurations
- Environment properties
- Intermediate and final performance equations

The database can store an unlimited number of different application case studies including the performance data necessary for analysis. Intermediate results are also stored in the database such that an interactive session can be saved/restored at any time. The database should contain "pointers" to the application software - the actual source code may be stored elsewhere.

We investigated suitability of O-O databases like Ontos, O2, Jasmine and Versant. We evaluated factors such as Object-oriented support, Graphical Development Tool, Web Interface, Language support and finally price. We also investigated the need for a centralized facility for all inter-tool communication and access to the object-oriented database.

We decided that the object-oriented database Jasmine developed by the Computer Associates is suitable for our needs. Jasmine product is currently in BETA release and will be released shortly. If there are any problems in the product release, we have selected Versant as a back-up alternative for the object-oriented database.

B.7 Modeling Specification

The Hierarchical Performance Modeling (HPM) methodology was developed during Phase I and the proposed system for Phase II is based on this methodology. The system level, module level, task level and operational level modeling specifications were developed. The detailed specifications are not provided in this report, but will be compiled as a separate technical document for reference in Phase II.

B.8 User Interface

We investigated the user interface that is needed to communicate with various performance modeling tools. We decided that a graphical user interface (GUI) would be designed to connect the entire tool sets and facilities to a window-based operating system. We have an initial specification of the user interface which is based on the concept that our tool should be similar to other engineering design tools currently available to software and hardware engineers. We plan to develop this user interface using Visual C++ or JAVA during our Phase II work.

B.9 Open System Mechanism

We have developed an Open System Mechanism for our proposed system. At the center of the architecture for our tool is the intelligent communication shell. This component acts as a real-time communication mechanism that allows new tools to be added in a modular fashion. The intelligent communication shell will use a standardized protocol that will allow interfacing to a wide variety of standard COTS software products. Technologies like CORBA and ACTIVE-X were proposed as application programming interfaces that could implement such an Open System Mechanism. A detailed account of this Intelligent Communication Shell is described in the system proposed below.

B.10 Applications Environment

During Phase I we considered three different categories of parallel/distributed computing systems. These are A) Shared Virtual Memory Systems, B) Message Passing Network-based Systems; C) Multi-threaded applications on tightly coupled multiprocessors. For each one of these architectures, we are developing the necessary HPM to predict the performance of an application written for these architectures. We plan to identify individual military applications for each of these target platforms which will serve as case studies. The three paragraphs that follow briefly describe our view of the modeling effort for each platform.

B.10.1 Shared Virtual Memory Systems

For shared virtual memory type systems, the programming model is based on a virtual, associative, logically-shared memory such as used in LINDA. This model is different from the conventional shared memory model used by machines such as Sequent, etc. Therefore, our previous modeling approach was modified during the term of the Phase I period to represent the virtual shared memory model. The new model considers: a) the process of creating different threads, b) the data structures used, c) the organization of the virtual memory, d) implementation of Linda/Paradise operations, e) the allocation protocol used to distribute different processes to processors, and f) the process to process interaction. We expect this modeling effort to be the most significant of the three areas because it requires a modification of previous modeling approaches.

B.10.2 Network-Based Systems

This class of architectures has received a great deal of attention because of platforms like PVM and MPI which facilitate programming these systems. In either case, applications programs are based on a set of library routines embedded in procedural languages such as C and FORTRAN. Support is provided for process-management and communication via connectionless and connection-oriented message passing. To develop the required models, we are planning to model different library routines such as PVM_SEND and PVM_RECEIVE. We will build an execution time model for each routine. Each routine is decomposed into one or more layers of components. This continues until the operation level where models of primitive operations of the network and architecture are used.

B.10.3 Tightly Coupled Shared Systems

This application domain includes a range of system types which employ tightly coupled shared memory multiprocessors combined with a multi-threaded operating system or library. The intention is to develop a model structure for the major COTS multi-threaded packages such as Solaris/Sun, Data-Parallel C, Windows NT and JAVA threads. There is also a need to develop models of real-time operating systems such as SPOX, iRMX, VRTX and pSOS which also provide multi-threaded operation. Our current plan is to modify existing models (such as the Computational Structure Model) in order to support this target area.

B.11 Tool Set Selection

Infopike, Inc. and UCONN had proposed to DARPA that they would select a minimum tool set to accomplish the intended objectives. We have done a thorough study on the appropriate tools and

selected a set of tools and an object-oriented database. Details of these tool selections are described in the Architecture Assessment Environment we proposed as a Phase II SBIR work. The sections C and D describe this tool set selection in detail.

B.12 Product Evaluation

During Phase I work, we have investigated the strategies through which the proposed product can be evaluated. As we develop various pieces of the proposed tool, we will install them at UCONN's and Infopike's facilities to test and debug the tool. We will utilize UCONN's research staff and Infopike's engineering staff to thoroughly test the product. We will select several potential customer sites as beta testing sites for our product. We have already prepared potential customer list out of our Industrial Consultations.

B.13 Training

We explored various alternatives of including training as part of the product offering. Customer training will be very important for our product. We decided that detailed brochures and training programs will be created and technical seminars will be organized during Phase II. We also talked to Connecticut Innovations to explore opportunities of getting support for commercialization phase (Post Phase II). Part of this support will be used for product training.

We organized a special training program as a graduate course at UCONN in Spring '97. The objective of the program is to prepare a group of trainers in the use of our performance tools to our future customers.

The product evaluation and training is an on-going program during Phase II. We plan to develop a program in which Infopike and UCONN will jointly train the target customers in the usage of the product being developed.

B.14 Case Studies

We have done two case studies to construct performance models using the Hierarchical Performance Modeling (HPM) methodology. The two case studies show model construction methods applied in contrasting types of parallel systems. The first case study was conducted on a distributed real-time system in a commercial environment. The second case study was conducted on a shared memory multiprocessor in a research environment. The primary characteristics of the case studies are summarized in Table 1.

Property	RTS-1000	SPARCcenter 2000
Environment	Commercial	Research
Application Domain	Embedded Real-Time	Multi-user Server and High Performance
System Type	Distributed, Embedded Real-Time	Tightly-coupled Shared Memory
Processor nodes	Intel 80486 50MHz	SPARC 50MHz
Inter-task Communication Model	Message Passing via proprietary communications kernel and high-speed serial links.	SunOS 5.2, Solaris 2.2 Threads API using shared memory.
Algorithm Type used in Case Study	Communications Protocol using integer arithmetic	Numerical integration algorithm (floating point)
Topology of Machine	Point to point communication links, processing nodes arranged in a tree structure	Symmetric multiprocessor with global bus and global shared memory

Table 1. Summary of Case Study Features

B.14.1 RTS-1000 Distributed Application

The RTS-1000 is a real-time distributed multiprocessor machine used primarily for ultrasonic testing of railroad lines. The embedded component of the architecture consists of A/D conversion hardware and a number of embedded microprocessor boards. The user interface consists of two PCs which are used for data presentation, operation control and storage.

There are five microprocessor boards in the embedded system which are linked together in a parallel, pipelined fashion using high-speed serial links. The RP0 and RP1 nodes contain Intel 80960 microprocessors, while the ST0, ST1 and REC nodes normally contain Intel 486 embedded PC motherboards. The nodes communicate using an asynchronous message-passing model, implemented with a proprietary real-time communications kernel (software), software drivers and a serial interface card (hardware). In order to reduce communication contention as much as possible, each processor node is connected to its parent and child nodes with a dedicated point to point link which is contention free. The serial links implement the pipelined architecture, which is necessary in order to process the incoming ultrasonic data at high arrival rates.

During normal operation, ultrasonic data in the form of messages is transferred from child nodes to parent nodes via a message passing protocol on the serial links. Each node performs a filtering

function on the incoming ultrasonic data, and passes the results up to a parent node. Within each node, there are three tasks rtask, stask and ctask executing simultaneously.

The focus of this study is the interaction of these three tasks, and the resulting queuing properties of messages in the ST0, ST1 and REC nodes. Since the focus is on communications, the case study included a detailed analysis of the software used to implement the communications protocol. The study involved the message queuing influences, the delays caused by interrupts and the impact of different message arrival properties.

The communications protocol is implemented by the sendmsg and recvmsg functions, which are written in C and compiled with the Borland C compiler. The sendmsg and recvmsg functions are the basis for the stask and rtask tasks respectively. The computational component of the time cost, i.e. the ctask task, is modeled as a parameter with a designer-specified mean and variance.

In the RTS-1000 system, the primary real time constraint is the time it takes to process a message through the pipeline. For the purposes of this case study, the goal is to predict this response time with some degree of accuracy. Thus the study focuses on the response time calculation of the system level queuing model. Note however that unlike most queuing models, this model is parameterized by all the lower levels of abstraction. The system level prediction is verified against measured execution times on the actual system. Untuned prediction accuracy within this domain fell into the range of 20%-30% of the measured value for a wide range of workload scenarios. This range of accuracy is typical for complex queuing models. The generated equations show the relationship between the primary parameters, namely message length, inter-arrival time, computational load and communication port overhead.

B.14.2 SPARCcenter 2000 Parallel Application

The SPARCcenter 2000 is a 12 processor tightly coupled shared memory multiprocessor. Symmetric multiprocessing is provided by a multi-threaded execution model using SunOS version 5.2 and Solaris 2.2 threads. This system's normal function is to carry out server duties and to run processes for multiple users. For the case study, the influence of these "interference" factors was eliminated by operating the system in a single user mode. Without interference factors, one application can use the full processing power of the machine by binding each thread to a processor through a light weight process.

The case study application is a parallel numerical algorithm that computes an integral using the Romberg integration technique. The program was implemented with the SunOS C compiler and Solaris threads library - In order to parallelize the application, the master-slave model was employed to decompose the domain of the problem into a smaller set of problems. A "worker" function was defined which executes as a set of slave threads, each slave thread taking an equal fraction of the total computation. The function of each worker is to compute the Romberg integration value over part of the domain. After each worker finishes, the workers synchronize, and then the master thread sums up the results from each to compute the numerical value of the integral.

For the purposes of this case study as an example of hierarchical modeling, the concern is the ability to predict performance. Thus the study focuses on the verification of model results against measured execution times. Computation of other metrics, such as predicted speedup, is a simple effort once the model has been verified to give accurate results. Untuned prediction accuracy within this domain fell in the range of 10%-14% of the measured value for a wide range of workload scenarios.

c. Results of Phase I:

The following is a list of technical accomplishments during phase I.

1. The results of the Phase I FEASIBILITY STUDY indicate both a positive interest in hierarchical performance modeling, and a clear path to design/implementation of a performance prediction toolset. During Phase I the Uconn-Infopike group presented our current proposal to representatives from several companies including: IBM T.J. Watson Research, Digital Equipment Corporation, Analysis and Technology, United Technologies Research and Pandrol Jackson. Feedback from the industrial contacts during Phase I indicate a high demand for performance modeling for time cost prediction in both the high-performance computing domain (Digital Equipment Corporation, Analysis and Technology, IBM) and in the real-time embedded domain (United Technologies Research Center, Pandrol Jackson). The resulting Marketing strategy is to deliver this product by three access methods: 1) as a PC-local or standalone product, 2) as a UNIX based client-server product; 3) As a internet based product which can be accessed remotely and uses a server to execute commands. This plan relies upon the implementation of the entire system in a programming language and development environment that supports cross-platform translation with minimal code changes. We have selected all supporting tools (i.e. Jasmine, Mathematica, Java, Visual C++) based on this principle.

2. Specification of the OBJECT-ORIENTED DATABASE is at the core of all other technical activities, and must be very carefully defined. We have selected Jasmine as the primary candidate for to support an object-oriented database to store performance information for the toolset.

3. Specification of the properties of the HIERARCHICAL PERFORMANCE MODELING must also occur, since this area will be referenced by virtually all software tools. During Phase I we outlined a complete specification for the inputs, outputs and functionality of the required hierarchical structure.

4. Specification/selection of a GRAPHICAL USER INTERFACE must occur since software system input definition is always needed. In this area, we have decided to model the graphical interface based on commonly used engineering CAD tools, such as windows-based compilers, schematic entry tools, VHDL simulators, etc. This approach will allow for a quicker acceptance of our product into the marketplace because the designer will be familiar with the graphical manipulation methods and menus. One primary aspect of this interface is the combination of windows based frames and hierarchical object presentation. Our graphical interface will use the standard, cross platform class libraries provided by Visual C++.

5. Selection of a small set of APPLICATION FOCUS AREAS will allow a realistic completion and delivery of a functional system during phase II. We have selected three primary focus areas: a message passing, distributed environment (PVM or MPI); a shared virtual memory environment for a network cluster of machines; and a symmetric multiprocessor environment, specifically SUN/SOLARIS multi-threading, WINDOWS NT multi-threading or real-time embedded applications running on tightly coupled processor architectures.

6. Specification of an initial TOOL SET, and the properties of each tool, to correspond to the above concerns. In this area, we have derived a complete set of specifications that will allow individuals to design and implement complete tools to support each level of the hierarchical

model. Some initial design work on the final tools has also proceeded based on these specifications.

D. Proposed Phase II System

D.1 Phase II Technical Objectives

In Phase II several of the technical objectives of Phase I will be carried out to the implementation stage. The work to be done in Phase II is intended to enable Infopike, Inc., to extend the current prototype HPM (Hierarchical Performance Modeling) tools from the University of Connecticut into a form which would be a marketable software product. The current prototype tools support the lower layers of the total hierarchy, and Infopike will work with UCONN as a part of the Phase II effort to carry out the tasks discussed briefly below. This will allow Infopike to develop a full hierarchical analysis system. For Phase II, the following tasks are planned:

1. Development of an OPEN AND FLEXIBLE FRAMEWORK for construction of the performance tool.

2. EXTENSION OF HPM (Hierarchical Performance Modeling) research results into the design of the commercial tool.

3. Design of an OBJECT-ORIENTED DATABASE to support the storage of complex performance data.

4. Design of the GRAPHICAL USER INTERFACE (GUI) for the performance tools.

5. Development of modeling for three types of SPECIFIC ARCHITECTURE ENVIRONMENTS

6. Case studies and evaluation based on several MILITARY and COMMERCIAL APPLICATIONS.

D.1.1 Design of an Open Tool Framework

This technical objective incorporates the development, design and implementation of an open and flexible high-level design of the performance tools to incorporate the GUI, object-oriented data base, model evaluation sub-tools and numeric/symbolic mathematical processing within a cross-platform development system. This open framework will allow standard COTS software components to be added in a modular way as the system evolves and new modeling tools and abilities are required.

D.1.2 Extension of HPM Results

The extension of HPM results from the University of Connecticut into a usable form within a commercial tool requires investigation and development in the following areas:

1. Development of model evaluation sub-tools for analysis at each level of the hierarchy.
2. Development of support linkages between modeling layers (to aid the automation of hierarchical model generation).

3. Development of functions for processing designer specification uncertainty in parameter values.
4. Development of all required performance evaluation measures and metrics.
5. Development of sensitivity calculation functions.
6. Development of time cost analysis and parsing of FORTRAN and C.

D.1.3 Design of the Object-Oriented Database

During the design process, it is necessary to retain the structure of the overall system and the relation that each sub-system has to the whole system. The designer must be free to move from one level to another during the design process and the system must be able to track this movement. The designer will, at different points in the design, wish to modify portions of the system and evaluate how these changes impact both the functional and performance of the complete system. It is necessary, also, to record design changes and variations.

Because of the above reasons, a database support system is needed to keep the design configuration information and provide facilities for accessing this information as needed during later stages of the design effort. The database support system should make use of the advances in the design of the object-oriented database system. It should have interfaces with different design (functional and performance) tools and be able to answer different designer's queries.

D.1.4 Design of the Graphical User Interface

A graphical user interface (GUI) will be designed to connect the entire tool sets and facilities to a window-based operating system. We have an initial specification of the user interface that is based on the concept that our tool should be similar to other engineering design tools currently available to software and hardware engineers. Thus we intend to model our GUI after the tools that designers are already familiar with in their everyday activities. This philosophy will help to ease the designer into the performance modeling domain by providing a similar look and feel to what is currently be used for design tasks. The GUI will allow seemless access to the object-oriented database, and provide a hierarchical navigation ability that is similar to that found in many modern engineering design tools.

D.1.5 Modeling for Specific Architecture Environments

This task will look at three specific and common classes of real systems and extend and integrate the model needs and capabilities to insure that the final HPM prototype environment can fully support these computing contexts. The past work of the UCONN group has focused on the following languages: C, FORTRAN, and C/LINDA based on both single and multiple-processor, shared memory architectures. It is important to extend these prior results to include a more extensive set of parallel/distributed platforms. The three classes which will be incorporated in the initial tool offering are: a message passing, distributed environment (PVM); a shared virtual memory environment for a network cluster of machines; and a symmetric multiprocessor system using multithreaded capabilities, such as SOLARIS or WINDOWS NT. This task will be active throughout the two years of the project.

D.1.6 Military Application Case Studies

During the last year of the project two case studies were carried out with the intent of assessing the functionality and usefulness of the developed approach and environment. One of these systems was a commercial system and one was a university-based research system. The commercial case study was provided by Pandrol-Jackson RTS1000 system as a preliminary test case to assess our current results. The RTS1000 system is a distributed real-time system used for railroad track inspection (high-speed flaw detection and pattern recognition). The university-based system case study involved a numerical integration algorithm implemented on a SUN SOLARIS multi-threaded multiprocessor environment. For Phase II we will select two moderately complex defense systems, and plan to consult with DARPA and A&T in the selection and availability of an appropriate system for case study.

D.2 Design of an Open Tool Framework

The functional organization of the performance tool is shown in figure 2. The designer will work with the graphical user interface to indirectly invoke each of the required tools and facilities to construct and analyze a hierarchical model. Performance information and the hierarchical structure are stored in the object-oriented database. Mathematica support is required at various points for symbolic math processing or generation of plots and graphs.

There are ten major components of the open and flexible tool architecture shown in figure 2.

Infopike needs to do the following tasks to complete the system shown:

- Develop the Performance Sub-tools QTED, TTED, MTED, and PAT
- Develop the Graphical User Interface and Facilities 1-3 (See next section)
- Develop the Object-Oriented Database
- Develop the Intelligent Communication Shell

The intelligent communication shell is the primary vehicle used to implement the open framework, and is elaborated on further in the next section.

D.2.1 Development of the Intelligent Communication Shell

At the center of the architecture for our tool is the intelligent communication shell. This component acts as a real-time communication mechanism that allows new tools to be added in a modular fashion. The intelligent communication shell will use a standardized protocol that will allow interfacing to a wide variety of standard COTS software products. As part of this technical objective, Infopike will develop the communication shell using the appropriate protocols to support the exchange of information among all components. Infopike will decide on which mechanisms can be used to communicate, and some preliminary decisions have already been made as specified below.

In the design of our high-level tool architecture and database interface, we considered the type of interface standards that database products claim to adhere to. Currently there are two de-facto standards for open and flexible object oriented communications. They are CORBA and ACTIVE-X. CORBA is the standard being promoted by the Object Management Group and ACTIVE-X

(Formerly OLE) is the Microsoft standard. Any database that is utilized for an open framework requires interfaces to both CORBA and ACTIVE-X. Our system will be built by making calls to application programming interfaces (APIs) which are CORBA compliant and ACTIVE-X compliant.

Fortunately, major object-oriented databases like Jasmine and Versant are both CORBA and ACTIVE-X compliant. JASMINE has built-in power of WEB functions that will be needed to implement an Internet based product delivery mechanism. JASMINE is currently the best suited to our object-oriented data base requirements, however the use of CORBA and/or ACTIVE-X will allow us to decouple the tools from the database. As an alternative, the VERSANT database will be employed if there are problems with JASMINE. Coming back to the interface with Mathematica, we will be able to interface either database to Mathematica via C++ "wrappers" which allow any of the tools to invoke Mathematica functions through the communication shell.

Another key to the flexible structure is the development of three "facilities" which allow the user to perform different functions with the system. Facility 1, Model Development, will specify a format for an input language for describing queuing networks and other graph based performance models. Facility 2 will require an interface for input of parameters of each workload scenario. There are also several different types of parameters required at each of the four modeling levels. Facility 3 will provide analysis of the hierarchical model including uncertainty (High uncertainty, Medium uncertainty, Low uncertainty), complexity reduction, conditional upper bounds, conditional lower bounds, average, variance, and other metrics based on the workload scenarios specified with Facility 2.

D.3 Extension of Hierarchical Performance Methodology (HPM)

Parallel/distributed systems are normally composed of four or more abstraction levels, which correspond to logical boundaries in the design of hardware, application software and operating system services. The levels are shown in the following Figure 3.

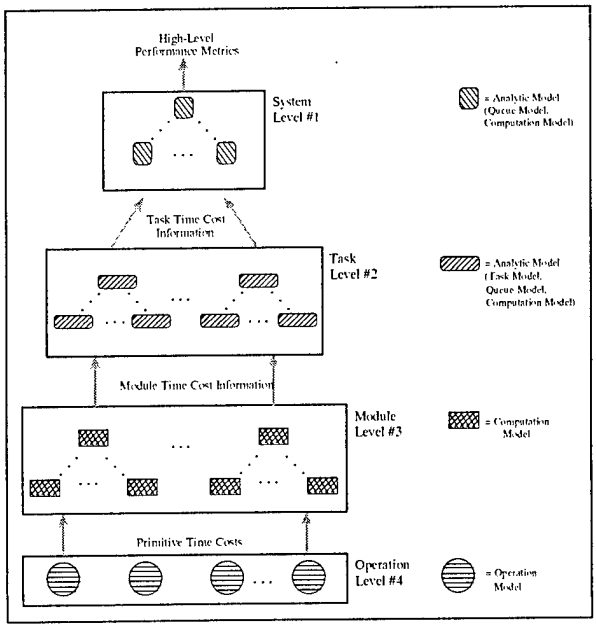

Figure 3. Hierarchical Performance Model (HPM)

To classify different types of performance models, a parallel/distributed system is decomposed into a series of modeling levels which are called the 1) System Level; 2) Task Level; 3) Module Level; 4) Operation Level. The approach used here is to match the design abstraction levels with the most appropriate set of "base" analytic models at each level of abstraction. The primary challenge then becomes how to link the base models at different levels together such that performance information can be propagated through the structure.

The operation level provides the lowest level of abstraction for computer performance modeling. The operation level estimates the time cost of primitive operations (such as addition, multiplication, assignment, etc.) which are contained in a software module. This level of analysis is determined by the high-level language compiler and microprocessor combination which establishes primitive time costs. The module level resides above the operation level in the series of levels. The module level calculates the time costs for segments of the software program (e.g. procedures or functions) based on the structures specified by the source code of a high level language. At the task level, the analysis is based on the interaction of software modules, which are running concurrently. At this level the system is viewed as a set of independent software units (e.g. tasks or threads) which synchronize and communicate in order to complete a larger job. Since the interaction between tasks changes the expected execution time of a software module, the task level analysis must provide estimates of the slowdown caused by interrupts and context switching. System level analysis describes a parallel or distributed application at the highest level of abstraction, utilizing coarse-grained estimates of the most significant service times, arrival rates and resource properties. Frequently the complete computer system is modeled as a single entity such as a queuing network. Queuing models become necessary whenever a contention for resources causes tasks to wait in a buffer for service. Based on these four levels of abstraction, a HPM approach provides a means to construct a complex but manageable hierarchical model.

D.3.1 Design of the Hierarchical Performance Sub-tools

The operation level tool is used to compute time estimates for operation level primitives. We currently have an initial version for this tool which is called PAT (Primitive Assessment Tool) that automatically calibrates an operation level model to fit the selected compiler/processor environment by timing all the required logical primitives. The module level tool (MTED) derives time cost expressions from the provided source code and flow parameter information. The inputs are application source code; flow parameter definitions in source code; and the table of logical primitives from the operation level tools. The task level tool (TTED) provides analysis of task models to calculate the effects of interrupts, context switching, scheduling and communication on task execution time. The system level tool QTED provides analysis of queuing models for both open and closed models, using exponential and non-exponential approximation methods.

Some ASAP modules such as TED and PAT will be ported to the new development environment using C++ "wrappers". TED will be developed into MTED, PAT will be enhanced to accept data from other operation level measurement tools such as Intel's VTUNE. QTED and TTED will be developed from the specification stage (no previous tool) using Mathematica calls for symbolic processing.

D.3.2 Hierarchical Linkages and Information Processing

This sub-task allows a performance model to be processed in a full hierarchy within the tool-set. We summarize the approach briefly for the subtask areas:

- Incorporate methods to link different types of base analytic models into a composite model.
- Incorporate methods for processing information with different levels of uncertainty.
- Incorporate techniques to evaluate the sensitivity of the performance equations.
- Incorporate methods to control the complexity of the performance equations.
- Integrate the methods as functions available to the designer within the framework of the performance toolset.

These areas constitute the mathematical foundation of HPM required to develop an initial commercial performance tool. The area of base model definitions is the starting point of the work in which the parameters and requirements for the basic models at each level are defined. The ability to link different types of models allows us to create macro models out of smaller models. For example, a queuing model may require several service time estimates that are provided by computation structure models. The area of sensitivity and complexity reduction involves the mathematical analysis and simplification of time cost equations. Finally, a methodology is required to integrate the previous areas together into a standard set of methods that can be used on a wide variety of applications.

D.3.3 Development of Multiple Language Base

This area of work will help provide for modeling multiple languages, specifically C, and FORTRAN. UCONN has previously developed two versions of prototype tools which support C and FORTRAN: 1) a FORTRAN-based ASAP tool for the Technical University of Munich where our approach is being applied to the analysis of Computational Fluid Analysis codes; and 2) a ANSI C-based ASAP [38], developed for Pandrol-Jackson. Infopike will start their development by extracting the parser sections of these tools, and adapting them to the time cost derivation tool MTED. Some additional debugging and fine-tuning will be required to insure commercial quality is achieved however significant re-design of the parser sections should not be required. Thus Infopike will be able to offer both C and FORTRAN source code analysis within the toolset without significant new development of parsers.

D.4 Design of the Object-Oriented Database

During the design process, it is necessary to retain the structure of the overall system and the relation that each sub-system has to the whole system. The designer must be free to move from one level to another during the design process and the system must be able to track this movement. The designer will, at different points in the design, wish to modify portions of the system and track how these changes impact both the functional and performance of the complete system. It is necessary, also, to record design changes and variations.

Because of the above reasons, a database support system is needed to keep the design configuration information and provide facilities for accessing this information as needed during later stages of the design effort. The database support system should make use of the advances in the design of the object-oriented database system. It should have interfaces with different design (functional and performance) tools and be able to answer different designer's queries.

D.4.1 Development of Primary Database Operations

The object-oriented database will provide a centralized facility for storage of all performance-related data. The primary database operations to be developed are:

- Operations for manipulation and representation of the Hierarchical structure of the model
- Operations for manipulation of sub-model data at each level (graphs, tables, time cost expressions, etc.)
- Operations for storage and retrieval of Parameters, measurement data, statistical information.
- Operations for storage and retrieval of Hardware configurations
- Operations for storage and retrieval of Software configurations
- Operations for storage and retrieval of Environment properties
- Operations for storage and retrieval of Intermediate and final performance equations
- Operations for storage and retrieval of final performance equations for future re-use

The database can store large number of different application case studies (projects) including the performance data necessary for analysis. Intermediate results are also stored in the database such that an interactive session can be saved/restored at any time. The database should contain "pointers" to the application software - the actual source code may be stored elsewhere.

The database support system should be designed to: A) Identify the appropriate modeling and analytical techniques to evaluate various design alternatives at different levels. B) Record the performance metrics at different levels and the role of each metric in the design process. C) Represent the interdependencies between the performance metrics used at different levels. D) Integrate these interdependencies into a system that is able to propagate any changes in a metric at a level to all upper levels. This allows the designer to evaluate his previous design decisions and their sensitivity to environment variations.

D.4.2 Development of a Homogeneous Tool-Interface

Since all the HPM sub-tools (PAT, MTED, TTED, QTED) require the object-oriented database as a central storage facility, the database must be easily accessible from a common interface. This approach will eliminate the discontinuity between different tools allowing data to be transported from one tool to another. We call the common interface outlined the "Homogeneous Tool Interface." The Homogeneous Tool Interface enables different steps in the evaluation technique to be implemented in separate performance tools and the results of one tool can be used as input for another tool.

JASMINE, an object-oriented database, was tentatively selected for support of the Homogeneous Tool Interface. JASMINE has several properties which are key to the implementation of the Homogeneous Tool Interface. They are:

- JASMINE supports multimedia applications that can be deployed on Intranets, client-server systems and the World Wide Web

- Applications created by JASMINE can be accessed remotely through a multi-platform web-based runtime environment

The Homogeneous Tool Interface will support object definitions which are independent of the application tools using the database. This approach will ease the extension of the system when new tools are developed.

D.5 Design of the Graphical User Interface

A graphical user interface (GUI) will be designed to connect the entire tool sets and facilities to a window-based operating system. We have an initial specification of the user interface which is based on the concept that our tool should be similar to other engineering design tools which are currently used by software and hardware engineers. This philosophy will help to ease the designer into the performance modeling domain by providing a similar look and feel to what is currently be used for design tasks. This GUI will allow seamless access to the object-oriented database, and provide a hierarchical navigation ability that is similar to that found in many modern engineering design tools. The GUI will provide a standardized menu system, with FILE, EDIT, VIEW, INSERT, TOOLS, WINDOW, and HELP menus. In addition, the GUI will allow for setup and tracking of multiple PROJECTS, so that many different application models can be stored in the same database. The hierarchical navigation ability is provided by a two-frame window in which the left frame shows the hierarchical structure of the model, and the right frame displays results and equations or allows input of data/graphs depending on the object selected in the left frame, the modeling level and the current context. Within the TOOLS menu, the designer can select from a number of choices:

- Drawing and Input of task graphs, CSM graphs, Queuing model graphs
- Check the completeness and correctness of created models
- Creating plots and charts of equations
- Statistical analysis (mean, variance, min, max, distribution information)
- Uncertainty analysis
- Sensitivity analysis
- Operating Region analysis
- Model development at each level
- Parameter and performance data Input

The FILE menu will allow for input and output of model files and for definition and editing of project setup files. The EDIT menu will allow for copy, paste, cut, delete, find and replace type operations which can be applied to named models at different levels or individual equations. The view menu will allow for different views of the hierarchical model, such as a detailed view in which all models are displayed in a graphical representation, vs a simplified view in which models are displayed as a "file list" using + and - signs to collapse and expand the hierarchy. The WINDOW menu will allow the design to have multiple concurrent activities in separate windows, such as simultaneous editing of a model structure at the system level while generating a plot of an equation in another window.

Major design and implementation tasks for Infopike are as follows:

- Design the three facilities (model generation facility, Input parameter processing facility, outcome analysis facility) which provide the functional capabilities to the GUI front end. This will be done using a combination of Visual C++ and Mathematica support calls.

- Designing interface between the GUI control center and the three "Facilities" using Visual C++;

- Design of a flexible control loop or control center for processing user input and generating outputs which is device independent and allows adding new tool interfaces to be defined.

- Development of specific functions within the user interface for drawing models, presenting data and equations, editing, and project/file management;

- Development of the actual user interface front-end features such as menus selections, window formats, frames, toolbars, radio buttons, scroll bars, etc.

D.6 Modeling for Specific Architecture Environments

In this project we are considering three different categories of parallel/distributed computing systems. These are: A) Shared Virtual Memory Systems; B) Message Passing Network-based Systems; C) Multi-threaded applications on tightly coupled multiprocessors. For each one of these architectures we are developing the necessary HPM to predict the performance of an application written for these architectures.

D.6.1 Development of Models for Shared Virtual Memory Systems

For shared virtual memory type systems, the programming model is based on a virtual logically-shared memory shared among a processor set through software-based memory access and communication facilities. This model is different from the conventional shared memory model used by machines such as the Sequent, etc. Therefore, our previous modeling approach will need to be modified during the term of this project to represent the virtual shared memory model. The new model must consider: a) the process of creating different threads, b) the data structures used, c) the organization of the distributed memory, d) the way each of the memory operations are implemented, e) the allocation protocol used to distribute different processes to processors, and f) the process to process interaction.

There will be two levels of analysis necessary to develop such as model. The research work of the lower level is to generate a library of performance models and performance equations for the intermediate memory functions such as IN, OUT. This library is the real key to achieving an adequate performance analysis of a given application. In effect, this allows a two-level hierarchical approach in the modeling process in order to allow a more robust incorporation of special language functions, routines which themselves are implemented at a lower (e.g., assembler/machine) level.

The second level of the research work is to modify the underlying computation structure model to match the way in which the memory operations are used. Every memory operation is replaced by a call to the pre-developed library to produce the appropriate cost equation of this operation. This cost equation will be included in the overall cost equation of the application.

D.6.2 Development of Models for Network-Based Systems

This class of architectures has received a great deal of attention because of platforms like PVM and MPI that facilitate programming these systems. In either case, applications programs are based on a set of library routines embedded in procedural languages such as C and FORTRAN. Support is provided for process-management and communication via connectionless and connection-oriented message passing.

To develop the required models we are planning to model different library routines such as PVM_SEND and PVM_RECEIVE. We will build an execution time model for each routine. Each routine is decomposed into one or more layers of components. This continues until the operation level where models of primitive operations of the network and the architecture are used. For example, PVM packs each message into its internal message buffers before sending it out to the network. Therefore, the execution time of the send operation is the sum of the pack and the send times. Then the pack time will be expressed as a function of the message size, the packing coefficient, and the overhead cost. The send time is expressed in the same way. Coefficients of the execution time equations will be evaluated using a benchmark approach over two different classes of networks - Ethernet and ATM.

D.6.3 Development of Models for Tightly Coupled Shared Memory Systems

This application domain includes a range of system types which employ tightly coupled shared memory multiprocessors combined with a multi-threaded operating system or library. The intention is to develop a model structure for the major COTS multi-threaded packages such as Solaris/Sun, Data-Parallel C , Windows NT and POSIX threads. There is also a need to develop models of real time operating systems such as SPOX, iRMX, VRTX and pSOS which also provide multi-threaded operation.

There are a number of new features which must be developed in our models to support multi-threaded shared memory architectures. The task and module level tools must be extended to support FORK, JOIN, LOCK and UNLOCK operations, to provide the basic building blocks for modeling the synchronization and communication mechanisms. It will be necessary to develop approaches for modeling thread scheduling effects, thread creation, thread suspension, thread barriers and other thread synchronization types at the task level. At the operation level, extended modeling is required in the area of modeling cache memory effects which can be dominant factors in the overall performance of tightly coupled systems.

D.7 Application Case Studies and Evaluation

For the Application case studies, we will target the following two categories of applications:

· A High Performance Computing Algorithm from Analysis and Technology

· A Military Real Time Computing Algorithm from DARPA

Applications such as nuclear simulations and battle scenarios fall into high performance computing. After the end of Cold War and signing of Nuclear Test Ban Treaty, there is a high demand for high performance computing. The organizations are interested in scalable supercomputing and have an

immediate need to verify that their complex software can be speeded up. They are interested in doing performance modeling and analysis without running lengthy benchmark programs.

We plan to identify individual applications in these categories further. We plan to use these applications in Phase II to test the tool we plan to develop. One application should be a military application such as a real-time radar (Synthetic Aperture Radar SAR) or sonar processing algorithm, the other will be selected based on input from A&T. We will specify how the application will map onto one of the three possible system architectures.

E. Project Administration

The following members are actively involved in investigating this Small Business Innovation Research:

• Ramesh Reddi	Infopike, Inc.	(Principal Investigator)
• Paul Johnston	Infopike, Inc.	
• Savitha Keshavan	Infopike, Inc.	
• Prof. Reda Ammar	University of Connecticut	(CO - Principal Investigator)
• Prof. Howard Sholl	University of Connecticut	
• Prof. Brian MacKay	University of Connecticut	
• Four Graduate Students	University of Connecticut	

Infopike has entered into a sub-contract agreement with University of Connecticut for this project. We have stated this arrangement in the initial Phase I proposal.

From the start, we have been meeting weekly and discuss the project issues. We believe that we have made considerable progress through these meetings and the completion of the assigned tasks to the team members.

F. Future

This research work will engender techniques to develop design aids for system engineers involved in designing complex systems. The research will show insights in system optimization, performance modelling and analysis of systems and thus pave the way for rapid development of these systems.

During Phase II, we will design and implement a Performance Analysis System that will be used as a tool in designing the identified applications. The system will be based on the system architecture outlined here. An efficient implementation based on COTS (Commercial Off The Shelf) philosophy will be used to re-structure the system for various applications.